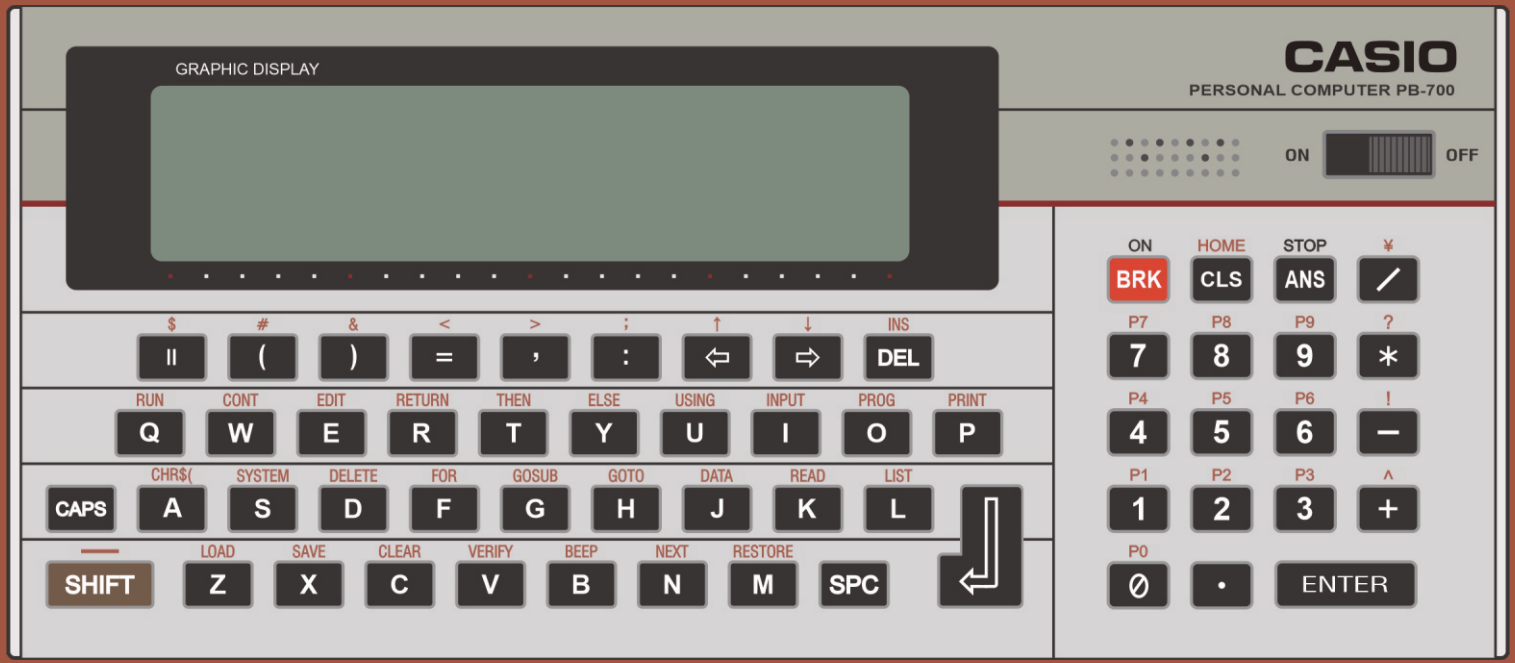
# CASIO

# PB-700


GRAPHIC DISPLAY
CASIO
PERSONAL COMPUTER PB-700
ON
OFF
$ # & < > ; ↑ ↓ INS
|| ( ) = , : ⇦ ⇨ DEL
RUN CONT EDIT RETURN THEN ELSE USING INPUT PROG PRINT
Q W E R T Y U I O P
CHR$( SYSTEM DELETE FOR GOSUB GOTO DATA READ LIST
CAPS A S D F G H J K L
LOAD SAVE CLEAR VERIFY BEEP NEXT RESTORE
SHIFT Z X C V B N M SPC
ON HOME STOP ¥
BRK CLS ANS /
P7 P8 P9 ?
7 8 9 *
P4 P5 P6 !
4 5 6 −
P1 P2 P3 ^
1 2 3 +
P0
0 . ENTER

## Command List

* Elements can be used repeatedly
{ } Select one of the elements
[ ] Elements can be omitted

## Manual Commands

### CONT

**Function:** Resumes the program which is in STOP status.
**Format:** CONT
**Example:** CONT

### DELETE

**Function:** Deletes a part of the program with a line unit.
**Format:** DELETE [starting line number] [-] [ending line number]
**Examples:** DELETE 50
DELETE 100-
DELETE -80
DELETE 70-120

### EDIT

**Function:** Displays the specified line and specifies the edit mode.
**Format:** EDIT [line number]
**Examples:** EDIT
EDIT 30

## LIST

**Function:** Lists the specified program.

**Formats:**
LIST ALL
LIST [starting line number] [-] [ending line number]
LIST V

**Examples:**
LIST ALL
LIST 20-50
LIST V

## LLIST

**Function:** Prints out the specified program.

**Formats:**
LLIST ALL
LLIST [starting line number] [-] [ending line number]
LLIST V

**Examples:**
LLIST ALL
LLIST 100-
LLIST V

## LOAD

**Function:** Loads the program on a cassette tape.

**Formats:**
LOAD ALL [“file name”]
LOAD [“file name”] [, {A or M}]

**Examples:**
LOAD ALL
LOAD “ABC”,A

## NEW

**Function:** Erases a program.
**Format:** NEW [ALL]
**Examples:** NEW
NEW ALL

## PASS

**Function:** Assigns a password to protect programs.
**Format:** PASS “password”
**Example:** PASS “ABC”

## PROG

**Function:** Specifies a program area.
**Format:** PROG numerical expression
**Example:** PROG 1

## RUN

**Function:** Executes a program.
**Format:** RUN [line number]
**Examples:** RUN
RUN 1000

## SAVE

**Function:** Stores a program on a cassette tape.
**Formats:** SAVE ALL [“file name”]
SAVE [“file name”] [,A]
**Examples:** SAVE
SAVE “MAY”, A

## SYSTEM

**Function:** Displays the program area status, number of remaining bytes and specified angle unit.
**Format:** SYSTEM
**Example:** SYSTEM

## VERIFY

**Function:** Verifies the program stored on the cassette.
**Format:** VERIFY [“file name”]
**Examples:** VERIFY
VERIFY “CASIO”

## Program Commands

### ANGLE

**Function:** Sets the unit for angle.
**Format:** ANGLE numerical expression (0<= numerical expression < 3)
**Examples:** ANGLE 0 (DEGREE)
ANGLE 1 (RADIAN)
ANGLE 2 (GRADIENT)

### BEEP

**Function:** Generates beep sounds.
**Format:** BEEP [{0 or 1}]
**Examples:** BEEP
BEEP 0
BEEP 1

### CHAIN

**Function:** Loads a specified program and executes it from the beginning.
**Format:** CHAIN [“file name"]
**Example:** CHAIN “TEST”

**CLEAR**

**Function:** Clears all variables.
**Format:** CLEAR
**Example:** CLEAR

**CLS**

**Function:** Clears the screen, and returns the cursor to the home position.
**Format:** CLS
**Example:** CLS

**DATA**

**Function:** Stores data to be referenced by READ statement.
**Format:** DATA data [, data] [, “character data”]*...
**Example:** DATA 5, 6, 8, 2

**DIM**

**Function:** Declares an array.
**Formats:** DIM array name [! or $] (numerical expression [, numerical expression])
DIM array name$ (numerical expression [, numerical expression]) * numerical expression
**Examples:** DIM A$ (10)
DIM L$ (5, 2)*3

## DRAW / DRAWC

**Function:** 1. Draws (erases) a point.
2. Draws (erases) a straight line.

**Formats:** DRAW (X-coordinate, Y-coordinate)[-(X-coordinate, Y-coordinate)]*...
DRAWC (X-coordinate, Y-coordinate)[-(X-coordinate, Y-coordinate)]*...

**Example:** DRAW (10, 5)-(20, 8)

## END

**Function:** Terminates program execution.

**Format:** END

**Example:** END

## ERASE

**Function:** Releases registered variables and array variables with a variable name unit.

**Format:** ERASE variable name [, variable name]

**Example:** ERASE A1 [, B1]

## FOR-TO-STEP / NEXT

**Function:** Repeats execution of the statements between the FOR statement and the NEXT statement a specified number of times.

**Format:** FOR variable name = numerical expression TO numerical expression [STEP numerical expression] NEXT variable name

**Example:** FOR I = 0 TO 9: NEXT I

## GET

**Function:** Reads variable data on the cassette.
**Format:** GET [“file name”] variable name [,variable name]*...
**Example:** GET A, B

## GOSUB / RETURN

**Function:** Causes a jump to a subroutine and a return from the subroutine, respectively.
**Formats:** GOSUB line number
GOSUB PROG program area number
RETURN
**Examples:** GOSUB 200
GOSUB PROG 3
RETURN

## GOTO

**Function:** Causes a branch to the specified destination.
**Formats:** GOTO line number
GOTO PROG program area number
**Examples:** GOTO 210
GOTO PROG 4

## IF-THEN-ELSE

**Function:** Performs a conditional branch, depending on the branch condition.
**Format:** IF conditional expression THEN
{line number (numerical expression) command}
(ELSE {line number (numerical expression) command})
**Example:** IF A>B THEN 10 ELSE 20

## INPUT

**Function:** Requests numerical value or character data entry from the keyboard.
**Format:** INPUT [“prompt statement”] {,;} variable name*...
**Example:** INPUT “DATA”;D

## LET

**Function:** Assigns data to a variable.
**Format:** LET variable name = numerical expression
**Example:** LET X = Y*2

## LOCATE

**Function:** Specifies the cursor position.
**Format:** LOCATE X-coordinate, Y-coordinate
**Example:** LOCATE 10, 2

## PRINT / LPRINT

**Function:** Causes an output to the display or printer, respectively.

**Formats:**
PRINT
PRINT expression [, expression]*...
PRINT “character string” [, “character string”]*...
LPRINT expression [, expression]*...
LPRINT “character string” [, ” character string”]*...

**Examples:**
PRINT “CASIO”
LPRINT A, B

## PUT

**Function:** Stores variable data on the cassette tape.

**Format:** PUT [“file name”] variable name [, variable name]*...

**Example:** PUT “SALES” A, B

## READ

**Function:** Reads data that was stored by the DATA statement.

**Format:** READ variable name [, variable name]*...

**Example:** READ S, T

## REM

**Function:** Gives a comment within the program.

**Format:** REM comment

**Example:** REM SUBROUTINE

## RESTORE

**Function:** Changes the execution order of the DATA statement.
**Format:** RESTORE [line number]
**Example:** RESTORE 1000

## STOP

**Function:** Temporarily stops program execution.
**Format:** STOP
**Example:** STOP

## TRON / TROFF

**Function:** Traces the program execution.
**Formats:** TRON
TROFF
**Examples:** TRON
TROFF

## Numerical Functions

### SIN

**Function:** Trigonometric sine function (sin X)
**Format:** SIN (numerical expression)
**Example:** SIN (A/B)

### COS

**Function:** Trigonometric cosine function (cos X)
**Format:** COS (numerical expression)
**Example:** COS (A*10)

### TAN

**Function:** Trigonometric tangent function (tan X)
**Format:** TAN (numerical expression)
**Example:** TAN (PI/6)

### ASN

**Function:** Inverse trigonometric sine function (arcsin or sin-1)
**Format:** ASN (numerical expression)
**Example:** ASN (X*X)

**ACS**

**Function:** Inverse trigonometric cosine function (arccos or cos-1)
**Format:** ACS (numerical expression)
**Example:** ACS (A+12)

**ATN**

**Function:** Inverse trigonometric tangent function (arctan or tan-1)
**Format:** ATN (numerical expression)
**Example:** ATN (A/100)

**EXP**

**Function:** Exponential function
**Format:** EXP (numerical expression)
**Example:** EXP (1)

**SQR**

**Function:** Square root
**Format:** SQR (numerical expression)
**Example:** SQR (30)

**LOG**

**Function:** Natural logarithm function
**Format:** LOG (numerical expression)
**Example:** LOG (2.71828)

**LGT**

| | |
|---|---|
| **Function:** | Common logarithm function |
| **Format:** | LGT (numerical expression) |
| **Example:** | LGT (100) |

**ABS**

| | |
|---|---|
| **Function:** | Gives the absolute value of the numerical expression. |
| **Format:** | ABS (numerical expression) |
| **Example:** | ABS (-10.5) |

**INT**

| | |
|---|---|
| **Function:** | Gives the largest integer which is less than or equal to the specified numerical expression. |
| **Format:** | INT (numerical expression) |
| **Example:** | INT (3.14) |

**FRAC**

| | |
|---|---|
| **Function:** | Gives the value of the fractional part of the numerical expression. |
| **Format:** | FRAC (numerical expression) |
| **Example:** | FRAC (2.64) |

**SGN**

**Function:** Gives the sign of the numerical expression.
**Format:** SGN (numerical expression)
**Example:** SGN (-1)

**ROUND**

**Function:** Gives the value obtained by rounding the specified digit.
**Format:** ROUND (numerical expression, digit position)
**Example:** ROUND (1.414, 2)

**PI**

**Function:** Gives the approximate value for circle ratio.
**Format:** PI
**Example:** PI

**RND**

**Function:** Gives a random number.
**Format:** RND
**Example:** R=RND

## Character Functions

### ASC

**Function:** Gives the character code of the character.
**Format:** ASC (Character expression)
**Example:** ASC (“A”)

### CHR$

**Function:** Gives the character represented by the specified code.
**Format:** CHR$ (numerical expression)
**Example:** PRINT CHR$ (62)

### VAL

**Function:** Converts a character string into the corresponding numerical value.
**Format:** VAL (character string)
**Example:** VAL (“123")

### STR$

**Function:** Converts a numerical value into the corresponding character string.
**Format:** STR$ (numerical expression)
**Example:** STR$ (123)

**LEFT$**

**Function:** Gives the specified number of characters from the left end of the character string.
**Format:** LEFT$ (character string, number of characters)
**Example:** LEFT$ (“ZYXWV”, 3)

**RIGHT$**

**Function:** Gives the specified number of characters from the right end of the character string.
**Format:** RIGHT$ (character string, number of characters)
**Example:** RIGHT$ (A$, 2)

**MID$**

**Function:** Fetches a specified character string from a character string.
**Format:** MID$ (character string, position [, number of characters])
**Example:** MID$ (A$, X, Y)

**LEN**

**Function:** Gives the length of a character string.
**Format:** LEN (character string)
**Example:** LEN (B$)

**INKEY$**

**Function:** Requests a single character entry from the keyboard.
**Format:** INKEY$
**Example:** K$ = INKEY$

## Other Functions

**TAB**

**Function:** Moves the cursor up to the specified number of character positions.
**Format:** TAB (numerical expression)
**Example:** PRINT TAB (10); 123

**USING**

**Function:** Specifies the display format.
**Format:** USING “format character string”;
**Example:** PRINT USING “##.###”; A

**POINT**

**Function:** Checks whether a specified point on the display has been turned on or off.
**Format:** POINT (X-coordinate, Y-coordinate)
**Example:** T = POINT (1, 3)

## Error Message List

**BS error** (**B**ad **S**ubscript)

1. A subscript of an array variable is a negative value, or there are more than 256 subscripts.
2. A specified subscript is outside the allowable range.

**BV error** (**B**uffer o**V**erflow)

1. Input buffer overflow.

**DA error** (**DA**ta error)

1 A READ or GET statement attempted to read non-existent data.

**DD error** (**D**uplicate **D**efinition)

1. The same array name with different subscripts was defined doubly.

**FC error** (illegal **F**unction **C**all)

1. An attempt was made to execute a manual command as a program command.
2. An attempt was made to execute a program command as a manual command.

**FO error** (NEXT without **FO**r)

1. There is no FOR statement corresponding to the NEXT statement.

**GS error** (RETURN without **G**o**S**ub)

1. There is a RETURN statement that is not corresponding to any GOSUB statement.

**MA error** (**MA**thematical error)

1. A mathematical operation or mathematical function operation is undefined or illegal.

**NO error** (**N**esting **O**ver error)

1. The nesting level is outside the allowable range.

**NR error** (device **N**ot **R**eady)

1. An I/O device is not properly connected.

**OM error** (**O**ut of **M**emory)

1. The RAM memory capacity is insufficient.

**OV error** (**OV**erflow error)

1. A result of an operation or input value is greater than 10^100.

**PR error** (**PR**otected error)

1. An attempt was made to execute a command which can not be used in a program protected by a password.
2. An attempt was made to add or delete a line to / from a program protected by a password.
3. The input password is incorrect.
4. An attempt was made to LOAD a program whose password is different from the main.

**RW error** (**R**ead **W**rite error)

1. A parity error occurred during execution of the LOAD or VERIFY command.

**SN error** (**S**y**N**tax error)

1. There is an error in a command format.
2. A fraction is included in a line number.
3. More than 3 dimensional array was declared.

**SO error** (**S**tack **O**ver error)

1. More than 8 levels of numerical value stack. The numerical value stacks exceed 8 levels.
2. More than 20 levels of operator stack. The operator stack exceeds 20 levels.
3. More than 10 levels of character stack. The character stacks exceed 10 levels.

**ST error** (**ST**ring error)

1. An attempt was made to assign a character string whose length is outside the allowable range to a character variable.

**TM error** (**T**ype **M**ismatch)

1. The variable types of the left side and right side in an assignment expression do not match.
2. The types of the assigned arguments do not match.

**UL error** (**U**ndefined **L**ine number)

1. The specified line number does not exist.
2. There is no program in the program area specified by the GOTO or GOSUB statement.

## **UV error** (**U**ndefined **V**ariable)

1. A variable which was not defined was used.
2. An array variable was used without the declaration by the DIM statement.
3. A subscript of an array variable exceeds the range specified by DIM.

## **VA error** (**VA**riable error)

1. An attempt was made to register more than 41 variables.

## **Statement**

| | | |
|---|---|---|
| Statement | Maximum number of characters per line | 79 |
| | Maximum line number | 9999 |
| FOR - NEXT nesting | | 6 |
| GOSUB nesting | | 12 |
| Multi-statement | | Possible |
| Outputs with specified format | | USING statement |

## Operations

| | | |
|---|---|---|
| **Arithmetic operators** | Addition | + |
| | Subtraction | - |
| | Multiplication | * |
| | Division | / |
| | Exponential calculation | ^ |
| | Surplus | MOD |
| **Relational operators** | Equal | = |
| | Not equal | < > , > < |
| | Less than | < |
| | More than | > |
| | More than or equal to | =>, >= |
| | Less than or equal to | =< , <= |
| **Priority of operations** | 1. Inside ( )<br>2. Functions<br>3. Exponentiation (^)<br>4. Signs (+, -) | 5. *, /<br>6. MOD (surplus)<br>7. +, -<br>8. Relational operators |
| **Computation and display** | Internal computation | Performed with 12-digit mantissa and 2-digit exponent |
| | Computation result display | Displayed with 10-digit mantissa and 2-digit exponent (rounded at the 11$^{th}$ digit of mantissa) |

# PB-700 Basic Specifications

| | | | | |
|---|---|---|---|---|
| Character processed | | Upper case and lower case letters, Katakana, and Special symbols | | |
| Contents | Numeral | Single precision | Number of significant digits | 12 |
| | | Half precision | Number of significant digits | 5 |
| | | Range of exponent | Maximum | +99 |
| | | | Minimum | -99 |
| | Character | Maximum character string length | | 79 |
| Variables | Variable name | Legal variable name characters | | Alphanumeric and symbols |
| | Numerical variables | Fixed numerical variable | Number of storable digits | 12-digit mantissa and 2-digit exponent |
| | | Registered numerical variable | | |
| | | Single-precision numerical array | | |
| | | Half-precision numerical array | | 5-digit mantissa and 2-digit exponent |
| | Character variable | Fixed character variable | Number of storable characters | 7 characters |
| | | Registered character variable | | 16 characters |
| | | Fixed-length character array | | 16 characters |
| | | Defined-length character array | | 0 ~ 79 characters |
| | Necessary number of bytes for an array variable | Single-precision numerical array | 8 bytes | |
| | | Half precision numerical array | 4 bytes | |
| | | Fixed-length character array | 17 bytes | |
| | | Defined-length character array | (number of stored characters + 1) bytes | |

<table>
<tr><td rowspan="6">Variables</td><td rowspan="3">Numerical variables</td><td colspan="2">Fixed variables (A ~ Z)</td></tr>
<tr><td colspan="2">Registered Variables (consisting of two upper case letters or numbers)</td></tr>
<tr><td rowspan="2">Array variables</td><td>Half precision numerical array<br>[ex. A! (i), A! (i,j)]</td></tr>
<tr><td>Single precision numerical array<br>[ex. A (i), A (i,j)]</td></tr>
<tr><td rowspan="3">Character variables</td><td colspan="2">Fixed variable (A$ ~ Z$)</td></tr>
<tr><td colspan="2">Registered variable (consisting of upper case letters, numbers, and $)</td></tr>
<tr><td rowspan="2">Array variables</td><td>Fixed-length character array<br>[ex. A$ (i), A$ (i,j)]</td></tr>
<tr><td>Defined-length character array<br>[ex. A$ (i) * 20, A$ (i,j) * 20]</td></tr>
</table>

## Plotter - Drawing Commands

| Command | Name | Function |
|---|---|---|
| O | ORIGIN | Defines an origin of ORG coordinate. |
| D | DRAW | Draws straight lines connecting the points specified by ORG coordinates. |
| I | RELATIVE DRAW | Draws straight lines connecting the points defined by the specified displacements in X and Y directions from the current pen position. |
| M | MOVE | Moves the pen holder assembly with the pen up to the point defined by the specified ORG coordinates. |
| R | RELATIVE MOVE | Moves the pen holder assembly with the pen up from the current pen position to the point defined by the specified X and Y coordinates. |
| A | QUAD-RANGE | Draws a quadrangle whose two diagonal points are defined by the two specified ORG coordinates and whose sides are parallel to the X and Y axes. |
| C | CIRCLE | Draws a circle or circular arc around the centre defined by the specified ORG coordinates. It draws an arc when the angle parameters are specified. |
| X | AXIS | Draws a coordinate axis in the +Y, +X, -Y, or -X direction from the origin of ORG coordinates. |
| G | GRID | Draws horizontal or vertical stripes from the current pen position within the specified range. |
| L | LINE TYPE | Specifies a line type which is solid line, broken line, one-dot chained line ot two-dot chained line. |
| B | LINE SCALE | Specifies the pitch of broken line, one-dot chained line or two-dot chained line. |

## Plotter - Character and symbol printing commands

| Command | Name | Function |
|---|---|---|
| S | ALPHA SCALE | Specifies the size of characters and symbols to be printed. |
| Q | ALPHA ROTATE | Specifies the rotational angle (orientation) of characters and symbols to be printed. |
| Z | SPACE | Specifies the spacing between the current and next characters and / or the spacing between the current and next lines. |
| Y | HORIZONTAL / VERTICAL PRINT | Specifies whether subsequent character strings are to be printed horizontally or vertically. |
| P | PRINT | Allows the specified character strings or data to be printed while in graphic mode. |
| N | MARK | Draws the specified mark centred at the current pen position. |

## Plotter - Control Commands

| Command | Name | Function |
|---|---|---|
| J | NEW PEN | Specifies the colour of pen; black, blue, green, or red. |
| F | LINE FEED | Feeds the paper by the specified number of lines. |
| H | HOME | Redefines the absolute coordinate system, or moves the pen holder assembly for inspection of the drawing. |
| @ | TEST | Allows trial drawing or a check for proper inking. |

## Plotter - Character control commands

| Command | Name | Function |
|---|---|---|
| **T** | **TAB** | Specifies a tab position. |
| **?** | **FORMAT** | Specifies a formatted program listing. |

- Graphic mode specification: LPRINT CHR$ (28); CHR$ (37)
- Character mode specification: LPRINT CHR$ (28); CHR$ (46)
- List format specification: LPRINT CHR$ (27); “?1"

## Character Code Table

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | (HOME) | (CLS) | (RET) | | |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| | (DEL) | (INS) | | | | (ANS) | (ENT) | (SHFT) | | | | (CSR>) | (CSR<) | (CSR^) | (CSRv) |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| (SPC) | ! | “ | # | $ | % | & | ’ | ( | ) | * | + | , | - | . | / |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| P | Q | R | S | T | U | V | W | X | Y | Z | [ | ¥ | ] | ^ | _ |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| ´ | a | b | c | d | e | f | g | h | I | j | k | I | m | n | o |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| p | q | r | s | t | u | v | w | x | y | z | { | ¦ | } | ~ | |

## Character Code Table

| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▁ | ▂ | ▃ | ▄ | ▅ | ▆ | ▇ | ■ | ▏ | ▎ | ▍ | ▌ | ▋ | ▊ | ▉ | ┼ |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| ┴ | ┬ | ┤ | ├ | ‾ | ─ | │ | ┆ | ┌ | ┐ | └ | ┘ | ╭ | ╮ | ╯ | ╰ |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| (SPC) | 。 | 「 | 」 | 、 | ・ | ヲ | ァ | ィ | ゥ | ェ | ォ | ャ | ュ | ョ | ッ |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| ー | ア | イ | ウ | エ | オ | カ | キ | ク | ケ | コ | サ | シ | ス | セ | ソ |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| タ | チ | ツ | テ | ト | ナ | ニ | ヌ | ネ | ノ | ハ | ヒ | フ | ヘ | ホ | マ |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| ミ | ム | メ | モ | ヤ | ユ | ヨ | ラ | リ | ル | レ | ロ | ワ | ン | ゛ | ゜ |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| 二 | 卜 | 丰 | 刊 | ◢ | ◣ | ◥ | ◤ | ♠ | ♥ | ♦ | ♣ | ● | ○ | ／ | ＼ |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |
| × | 円 | 年 | 月 | 日 | 時 | 分 | 秒 | 〒 | 市 | 区 | 町 | 村 | 人 | ▨ | |

CASIO

Digitally reconstructed by Steven Fox 2021